



FreeBSD uzerinde IpFilter kurulumu ve konfigurasyonu.

Giris

5 Haziran 2001

Bu makalede, daha once, dunyanin en guvenli Isletim Sistemi olarak, dost dusman butun guvenlik otoriteleri tarafından kabul goren, ve Bugtraq listesinde guvenlik aciklari en az boy gosteren bir isletim sistemi olmasiyla goz dolduran OpenBSD uzerinde efsanevi BSD paket filter'i enfes ve muthis derecede etkin ve etkili kullanan basarili bir `paket filter l` olan Ipfilter'in bir `production` firewall makinasi uzerinde kurulumu anlatilmisti.

Fakat, dort bes aydan beri bazi seyler degisti. Degisen en buyuk sey de, iki uc gun once Theo de Raadt'in, OpenBSD cvs tree den ipfilter'i cikarmasÄ± kuskusuz. Yani, onumuzdeki OpenBSD lerde ipfilter olmayacak. Bunun sebebi olarak da, Darren Reed'in ipf34-current icin (ipfilter'in development versiyonu, Darrenin deyimiyle non-release code) LICENCE 'i biraz degistirmesi yatiyordu. Darren, latest-release'de (3.4.17) eski BSD lisansini hala kullaniyor, degisikligin sadece, "release kabul edilmeyen surumun, kendisinden izinsiz 'degistirilmesinin' onune gecmek" amaciyla yapildigini, dolayisiyla, muhtemel breakdownlarin onune gecmek istedigini soyluyor. Fakat, OpenBSD takimi ve Darren bu konuda anlasamadilar, ve IPF, OpenBSD cvs tree'den cikarildi. Buna ragmen, Darren insanlari geri donut vermeleri durumunda OpenBSD icin destegini surdurecegini belirtti:

http://false.net/ipfilter/2001_05/0565.html

Darren, neden bu degisikligi yaptigini soyle anlatiyor:

http://false.net/ipfilter/2001_05/0458.html

Aslinda bu dokumanda TODO olarak, OpenBSD ipfilter bridging i anlatmak duruyordu, fakat, bu durumda bunu anlatmak pek feasible bir dusunce degil.

Fakat, Darren ve NetBSD ile FreeBSD 'core team' larinin yaptigi gorusmeler sonucunda, bu iki BSD isletim sistemi ipfilter'in kendi cvs tree'lerinde yer almasinda bir sakinca olmadigi gorusunde birlestiler. Darren'in bu konuda yaptigi aciklama maili asagida:

http://false.net/ipfilter/2001_06/0051.html

Her neyse devam edelim,

Eger, sanal ortamdan korumaniz gereken `top-secret` bilgiler tasiyan birkac yuz ile birkac bin arasi makinanız var, ticari firewall cozumlerini size kakalamaya calisan pazarlamacilarin istedigi kadar cok paraniz da yoksa, bu dokumani okumaya devam edin. FreeBSD+ipfilter tam size gore. Cunku siz `cok sey` istiyorsunuz ama, `az paraniz` var. ;)

Genel Notlar

Bu dokumanin en guncel hali; <http://www.enderunix.org/docs/ipfilter.html> adresindedir.

Aksi belirtilmedigi takdirde bu kabil dokumanlarin haklari kendilerini yazan yazarlarda saklidir. Bu dokuman da, parca parca ya da tamamen herhangi bir sekilde, yazarinin izni dahilinde dagitilabilir.

Yazar, bu dokumani okuyanlarin ugrayacaklari herhangi bir zarardan oturu sorumluluk kabul etmez. Use at your own risk!

Bu makalede, yeni kullanicilara ipfilter paketinin tanitimi, ve guzel firewall dizayni hakkında bir kac bilgi verebilmek amaciyla yazilmistir. Kullanicinin temel network kavramlari, temel isletim sistemi kavramlari, paket filtreleme, tcp/ip hakkında genel bilgisi oldugu kabul edilmektedir. Temel seviyede FreeBSD tecrubesi isinizi kolaylastirir, calisir bir sisteme kavusmak icin gerekli degildir, ama temel *NIX bilgisi gerekmektedir!!!. Fakat firewall'inizin isletimini devam ettirmek icin BSD ogrenmek zorundasiniz!!!.

Egerki, Linux uzerinde bir firewall uygulaması (ipchains'la) Ismail hoca'nin Ipchains-Howto'sunu okuyun efendim. Adresi:

<http://www.enderunix.org/docs/ipchains.html>

Burada, temel (www/sntp/ssh/pop/imap gibi) birkac servisin firewall kontrolu altinda alınmasi, ve organizasyonun internet cikisinin firewall'imizdan yapilmasi icin gerekli konfigurasyonu, ipfilter'i genel olarak anlatarak yapacagiz.

Burda DMZ ve trusted net arasinda herhangi bir kural konmamistir. Gene zaman darligi nedeniyle bu konu ileriki bir tarihe atilmistir.

Ayrica, kurallari, rulegroup'lar haline getirilmesi de burada anlatilamistir. Bu da ileriki bir tarihte yapilmaya calisilacaktır.

Fragmented paketlerin hepsi `ignore edilmistir`. Firewall'imiz fragmented paket kabul etmez. Isterseniz `keep frags` kelimesiyle burada da degisiklik yapabilirsiniz.

Eger, herhangi bir konuda yardima ihtiyaciniz olursa, roots@enderunix.org adresine mail atabilirsiniz.!

4 Haziran, 2001

FreeBSD mi OpenBSD mi?

OpenBSD'nin <http://www.openbsd.org/> de bulunan Turkiye yansi sayfasina girerseniz, gozunuz ilk carpan sey, sayfasinin ortasinda kirmizi harflerle yazilan su ibaredir: "**Four years without a remote hole in default install**". Yani bu su demek, 4 yil once kurdugunuz teknolojik olarak bile outdated olmus bir OpenBSD makinasi, eger tehlike arz edecek yerel kullaniciniz yoksa, remote olarak kale gibi guvenli. Makinayi default olarak kuruyorsunuz, 4 sene yanina hic gitmiyorsunuz, patch'lemiyorsunuz, iki de bir de bugraq'dan gelen bir mail'le irkilip gecenin bi yarisi ofise gidip de paket guncellemekle ugrasmiyorsunuz, makinaniz hala guvenli sayilabilir durumda oluyor. Hala `hacker-free`!! Burada guvenli derken, OpenBSD'de remote r00t acigi bulunmadigini ve kimsenin bulamayacagini iddia etmiyoruz. Dedigimiz sey, underground piyasasinin seferber olmasina karsin, kaynak kodu da, virgulune kadar acik oldugu halde hala `root` verecek bir remote bir acigin bulunamamasi, Isletim Sistemi tarihinde efsanevi bir olaydir.

Biz, organizasyon guvenligimizin merkezi durumunda olacak makinamizin, herseyin ustunde `sekuyr` olmasini isteriz. Oyleyse, herseyden once, onun isletim sistemi bizim icimize su serpmelidir. Baskalarini koruyayim derken, arkadan vurulmamiz icin, once firewall makinamizin kalbi beyni herseyi olan isletim sisteminin bizim basimizi agritmayacak kadar guvenli olmasi lazimdir.

OpenBSD'nin bu gibi artilari olmasina ragmen, GIRIS kisminde da belirttigim gibi, su anda OpenBSD gelistircileri official olarak ipfilter'i cvs tree'lerinde tutmuyorlar, yani destek vermiyorlar. Ama gene dedigim gibi, Darren feedbacklerin surmesi durumunda OpenBSD icin destegini devam ettirecegini soyluyor.

OpenBSD'yi nasil kuracagim?

Endiseye hic mahal yok! Omer Faruk Sen'in yazdigi OpenBSD kurulumu gayet aciklayici bir dokuman, zaten kurulumu tecube ettiginizde su ana kadar neden OpenBSD kurmadiginiza pisman olacaksınız. Simdi, http://www.enderunix.org/docs/openbsd_install.html yi okuyun, sonra burdan devam edin efetim...

FreeBSD ise, OpenBSD kadar, guvenlige adanmis bir gelisim surecinde olmamasina ragmen, FreeBSD'nin kurulumu yapilirken, sadece ve sadece 'core system'in kurulmasiyla ayni guvenlige sahip olabilir. Sadece 'core system' diyorum ve firewall olacak makinada acik portun olmamasi gerektiginden bahsetmiyorum bile!

Gelecegin ne getirecegini bilemeyiz, fakat, openipf.org domain adresi alinmis gorunuyor :)

Biz de simdi, ipfilter'i official olarak destekleyen iki BSD sistemden birisi olan, ve sunucu kurulumlarında robust olmasi ile un yapan FreeBSD uzerinde kuracagiz.

FreeBSD'yi nasıl kuracağız?

Gene endiseye mahal yok, EnderUNIX grubunun hazırladığı FreeBSD kurulum dokümanı var:
http://www.enderunix.org/docs/freebsd_kurulum/

Ipfilter

Ipfilter, Darren Reed (darrenr@pobox.com) tarafından yazılmış, firewall ortamlarında başarıyla uygulanan bir TCP/IP paket filtreleme programıdır.

BSD'nizin kernel'ına loadable module olarak yüklenebileceği gibi, statik olarak da derlenebilir, fakat tavsiye edilen static olarak derlemenizdir.

<http://avalon.coombs.anu.edu.au/~avalon/> adresinde yazıldığı üzere, Ipfilter,

- FreeBSD
- NetBSD

üzerinde kurulu geliyor, ve aşağıdaki platformlar üzerinde de derleyip çalıştırmışlar var:

- OpenBSD (daha yeni bir zamana kadar bu işletim sisteminde de default geliyordu)
- Solaris/Solaris-x86 2.3 – 8
- SunOS 4.1.1 – 4.1.4
- BSD/OS-1.1 – 4
- IRIX 6.2
- HP-UX 11.00 üzerinde de ipfilter 4.0alpha kullanılmış.

Eğer, paketi başka bir platformda denemek isterseniz, ipfilter'i

<ftp://coombs.anu.edu.au/pub/net/ip-filter/ip-fil3.4.17.tar.gz> adresinden indirip derleyebilirsiniz. Peki ipfilter neleri yapabiliyor?

1. paketi acık ve secık olarak gecirebiliyor yada bloke ediyor.
2. network arayuzlerini (network interface) ayirt edebiliyor.
3. ag/makina bazında filtreleme yapabiliyor
4. herhangi bir protokolu (tcp/udp/icmp gibi) filtrelebiliyor
5. bloke edilmiş paketler için, istenirse geri ICMP error/TCP reset paketi gonderebiliyor.
6. TCP, UDP ve ICMP paketleri için `packet state` yapabiliyor. İşte bu özelliği çok hoş. ve onu Linux'un ipchains'ından çok çok one geciren bir özelliği. Sonra görecez.

7. Ayni kurali butun fragmente paketlere uygulayarak herhangi bir IP paketi uzerinde fragment-state uygulayabiliyor. Nasil? gercekten hosunuza gitmeye basladi degil mi? ;)
8. Network Address Translator (NAT) yapabiliyor. Linux'ta IP-Masquaring bunun ismi.
9. Gercek transparent-proxy ortamlari icin `redirector` olabiliyor.
10. Port bazinda filtreleme yapabiliyor
11. Istenilen paketleri loglayabiliyor.

Iste bunlar Ipfilter'in onemli gordugum ozellikleri.

Simdi, kurmaya baslayalim... -->

On hazirlik

Once, firewall ortamimizi dizayn edelim: Bir internet agimiz var, bir adet DMZ yani server zone'umuz var, bir de trusted-net'imiz, yani pc'lerimizin felam oldugu bolge var. Yani toplam, birbirinden fiziksel katmanlari ayri 3 agimiz var:

Dolayisiyla, firewall olacak makinamizda 3 ethernet kartina ihtiyacimiz var. Bunlar, buyuk ihtimalle, hele hele GENERIC kernel kullaniyorsaniz acilis sirasinda asagidaki gibi taninacaktır:

```
fxp0: <Intel Pro 10/100B/100+ Ethernet> port 0x6400-0x641f mem
0xe4000000-0xe40fffff,0xe4100000-0xe4100fff irq 10 at device 10.0 on pci0
fxp0: Ethernet address 00:d0:c2:a0:a4:c1
fxp1: <Intel Pro 10/100B/100+ Ethernet> port 0x6400-0x641f mem
0xe4000000-0xe40fffff,0xe4100000-0xe4100fff irq 3 at device 10.0 on pci1
fxp1: Ethernet address 00:a0:c9:a0:a4:c2
fxp2: <Intel Pro 10/100B/100+ Ethernet> port 0x6400-0x641f mem
0xe4000000-0xe40fffff,0xe4100000-0xe4100fff irq 9 at device 10.0 on pci2
fxp2: Ethernet address 00:a0:c9:a0:a4:c3
```

Dolayisiyla ethernet arayuzlerimiz fxp0, fxp1 ve fxp2. bunlardan

- fxp0 internete bagli kartimiz, IP: 193.140.205.219
- fxp1 DMZ'e bagli kartimiz, IP: IP: 192.168.0.1
- fxp2'de pc'lerimizin bagli oldugu zone olacaktır. 172.16.0.1 --- 16 bit ag adresli.

Simdi de, makinamiz bir gateway olacagi, yani paketleri interface'leri arasinda forward edecegi icin, ip forwarding'i etkinlestirmemiz lazim, bunun icin, /etc/sysctl.conf'ta:

```
net.inet.ip.forwarding=0 satirini net.inet.ip.forwarding=1 olarak degistirelim. Eger bu dosya yoksa olusturun, ve icine net.inet.ip.forwarding=1 satirini ekleyin. Sonra makinayi reboot ediyoruz. Eger, reboot etmek istemiyorsaniz:
```

```
pathfinder:~# sysctl -w net.inet.ip.forwarding=1
```

```
net.inet.ip.forwarding: 0 -> 1
```

olarak degisecektir.

Baslatma

Simdi. FreeBSD'nizi basariyla kurdunuz,

```
pathfinder:~# netstat -a | grep LISTEN
tcp4 0 0 *.ssh *.* LISTEN
tcp46 0 0 *.ssh *.* LISTEN
tcp4 0 0 *.sunrpc *.* LISTEN
....
```

Evet, gordugunuz gibi, default olarak bir suru servisimiz hic gereksiz yere calisiyor. Bunun icin inetd yi disable ediyoruz.

```
pathfinder:~# kill -9 inetd_pid
```

FreeBSD'nin acilirken okudugu dosyalardan olan /etc/rc.conf u favori editorunuzle aciniz, Burada asagidaki satirlari ekleyin:

```
pathfinder:~# vi /etc/rc.conf
```

Orada, ipfilter ve ipnat'i calisir hale getirelim, gereksiz servisleri kapatalim:

```
ipfilter_enable="YES"
ipfilter_program="/sbin/ipf -Fa -f"
ipfilter_rules="/etc/ipf.rules"
ipfilter_flags="-E"
ipnat_enable="YES"
ipnat_program="/sbin/ipnat -CF -f"
ipnat_rules="/etc/ipnat.rules"
ipnat_flags=""
ipmon_enable="YES"
ipmon_program="/sbin/ipmon"
ipmon_flags="-Ds"
sendmail_enable="NO"
inetd_enable="NO"
```

```
usbd_enable="NO"  
portmap_enable="NO"
```

yapalım. Makinayı reboot ettiginizde, makina acilirken "*Configuring Ipfilter*" diyerek ipfilter'i calistiracaktır. Ya da, reboot etmek istemiyorsanız;

```
pathfinder:~# ipf -Fa -f /etc/ipf.rules -E  
pathfinder:~# ipnat -CF -f /etc/ipnat.rules -E
```

yapınız. burada `ipf` ve `ipnat` programlari filter ve nat rule'larini manipule eden baslica programlar oluyorlar.

Ama durun, hemen simdi reboot etmeyelim, cunku ipfilter'i kernel icine derlememiz lazim:

Kernel derleme

Ipfilter, FreeBSD 4.3 Release ile beraber geliyor, fekat kernel'da destegini vermek gerekiyor, Onun icin, /usr/src/sys/i386/conf/GENERIC ya da konfigurasyon dosyaniz neyse, ona minimum su iki satiri eklemeniz gerekiyor:

```
options          IPFILTER  
options          IPFILTER_LOG
```

Basitce, sonra:

```
pathfinder:~# /usr/sbin/config GENERIC  
pathfinder:~# cd ../../compile/GENERIC  
pathfinder:~# make depend  
pathfinder:~# make  
pathfinder:~# make install  
pathfinder:~# reboot
```

sonra, olarak kernel'i derleyeceksiniz. Daha detayli kernel derlemek icin de: <http://www.enderunix.org/freebsd/doc/handbook/kernel.html>

adresindeki Kernel Handbook'tan yardim alabilirsiniz...

Genel Mantik

Ipfilter, her rule'un eklenmesinde ayni binary'nin defalarca calistirilmasi yerine, calistiginda parse ettigi bir ruleset dosyasi kullanir. Ipfilterin paket filtelemesi, by-default, /etc/ipf.rules'deki ruleset'i, NAT'i da, /etc/ipnat.rules'dakileri okuyarak calisir. Simdi, /etc/ipf.rules dosyasini acalim, hem rule yazalim, hem de ipfilter'in calisma mantigini anlatalim:

Firewall kurallari, yukaridan asagiya okunur ve her yeni kural, ruleset'in sonuna eklenir.

```
pass in all
pass out all
```

Bir paket geldiginde, bilgisayar once, *pass in all* kuralini uygular, sonra da *pass out all*. Linux ipchains/iptables/ipfwadm ile ilgilenmis olanlar bilirler, orda bir paket geldiginde, kurallar yukaridan asagiya incelenir, eger paketi ilgilendiren bir kural yakalanirsa, ruleset'in taranmasi biter, ve o kural, nihai kural olarak pakete uygulanir.

Fakat, durum ipfilter'de durum boyle degil. Her paket icin, ruleset'deki butun kurallar kontrol edilir. Bu durum da *quick* kelimesiyle degistirilebilmektedir. Her neyse, devam edelim:

Yukaridaki kurallar her ingilizce bilen erkisinin anlayabilecegi uzre, gelene gidene eyvallah demektedir. yani gelen ve giden butun paketlere gecis izni vemektedir. Biraz daha ilerleyelim, peki bu napiyor?

```
block in quick from 192.168.0.0/16 to any
```

192.168.0.0 ag adresli, 192.168.255.255'den yayın yapan, yani 16-bit ag adresine sahip paketlerin gecisine, diger kurallari incelemeyi birakarak (*quick* kelimesiyle) izin vermiyor. Peki bu napiyo?

```
block in quick on tun0 from 192.168.0.0/16 to any
```

gene ayni sey gecerli, ama bu defa tun0 arabirimi uzerinde bu kural gecerli (tun0 bsd'de point-to-point baglantinin interface'idir). Peki bu?

```
block in log quick on tun0 from 192.168.0.0/16 to any
```

Bu da, bir onceki kuralla ayni etkiye sahip, ama bu sefer, bu paketi match eden paketleri logluyor.

```
pass in quick on tun0 proto icmp from any to 193.140.205.0/24 icmp-type 0
pass in quick on tun0 proto icmp from any to 193.140.205.0/24 icmp-type 11
```

proto icmp, kuralin icmp paketleri icin gecerli oldugunu berlitiyor. Himm, icmp paketinin tipini bile belirtebiliyorsunuz: ping ve traceroute calissin diye:)

```
block in log quick on tun0 proto tcp from any to 193.140.205.219/32 port = 513
```

tcp/udp paketleri icin port dahi belirleyebiliyoruz. makinamizin remote shell portuna giden paketleri bloke ediyoruz simdi.

FreeBSD IpFilter Howto

Simdi, Cisco kullananlariniz heman hatirlayacaksiniz, cisco'da `established tcp` paketlerinin gecisini saglayan established kelimesini. ipfw'de established, ipfwadm'de setup/established' dir o hani? Iste, ipfilter, bu established baglantilarin gercekten onlar mi olduklarini gercekten cok iyi takip edebiliyor. Hem de, sadece tcp bazinda degil, udp ve icmp bazinda da! Bu fonksiyonellik icin kullanmaniz gereken sihirli sozcuk:

```
keep state
```

Simdi, biraz olayin felsefesine girelim. Istemeyenler bir sonraki paragraftan devam edebilirler. ipfilter'da, gelen ve giden paketler icin aslinda, ruleset kontrol edilmez. Aslinda kontrol edilen `state table` dir. bu tablo, firewall'dan sorgusuz sualsiz gecmeesine izin verilen tcp/udp/icmp sessionlarin tutuldugu bir tablodur. Bir nevi bir guvenlik acigi gibi gorunse de aslinda degil. Neden mi? Butun tcp/ip baglantilarinin bir baslangici, bir devami bi de sonu vardir. Yani ortasi olmayan bir son'lu baglanti olmaz! ya da baslangici olmayan!. Eger, firewall kurallarimiz, bir baglantinin baslangicina izin veriyorsa, mantiki olarak, ortasina ve dahi sonuna da izin verecektir, degil mi? Iste, keep state olayi, firewall'in sadece yeni (diger bir deyimle SYN flag'i set edilmiş) paketler uzerinde islem yapmasini ve orta ve sonla ugrasmamasini sagliyor. Eger bir baglanti'da SYN paketinin gecmesine izin veriliyorsa, sonra gelen ACK ve FIN paketlerine de izin veriliyor. Iste state-table bu izin verilen baglantilardan olusuyor. Keep state bize gelen her incoming baglanti icin, outgoing kurallarimizda bir `pass out ...` kurali eklemekten kurtariyor. Oyle ya, bi taraftan gelen bir paket, ayni zamanda bizden oraya ulasabilmeli. Incoming olarak bi pakete izin veriyorsak, ondan kaynaklanan outgoing paketlerde otomatik olarak state table'a yaziliyor.

Simdi de, FIN saldirilarini, bu state tablosu yardimiyla, ve de `flags S` kelimesi yardimiyla engelleyelim. Bu kelime ustunde SYN flag basili paketler demektir;

```
pass in quick on tun0 proto tcp from any to 193.140.205.1/32 port = 23 flags S keep state
```

Simdi, 193.140.205.1 makinasina sadece SYN flagi basili paketler izin veriliyor, yani, kimde bize direk olarak FIN paketi yollayamaz, once SYN le established bi baglantisinin olmasi lazim.

```
/etc/ipf.rules
```

Eveet, simdi firewall kurallarimizi yazmaya baslayalim:

Simdi, iki cesit firewall dizayni var.

1. Ontanimli olarak, butun paketlere izin verip, iclerinden istediklerimizi yasaklamak.
2. Ontanimli olarak, butun paketleri bloke edip, iclerinden istediklerimize izin vermek.

Efendim, bendeniz, `security through obscurity` taraftariyim, onun icin 2. secenegi uygulayacagiz ;)

```
pass out quick on lo0
pass in quick on lo0
```

FreeBSD IpFilter Howto

loopback arayuzune gelis-gidis izni verelim. lo0 makinanın kendisi oldugu icin bu kural zorunlu!

```
block in quick on fxp0 all with opt lsrr
block in quick on fxp0 all with opt ssrr
```

Bazi sacma sapan paketlerin rahatimizi bozmamasi icin bunu ekliyoruz. bkz: <http://www.enderunix.org/openbsd/faq/faq6.html>

```
block in quick on fxp0 proto tcp from any to any flags FUP
```

Nmap gibi programların `tcp-ip fingerprinting` yontemiyle agimizdaki makinalarin isletim sistemini tespit etme amaclarini bu kuralla onluyoruz.

```
block return-icmp(port-unr) in quick on fxp0 from any to any port = 113
```

ident portuna erisim oldugunda, portumuzun kapali oldugunu port unreachable icmp paketi ile iletiyoruz. Bu, sunun icin gerekli: mesela bir makina telnet baglantisi aciyorsunuz. Karsi makina, sizin 113 portunuza baglanmaya calisacak, orda calismasi *gereken* identd ile konusmak isteyecektir. Ama firewall'da biz default olarak butun paketleri drop ettigimiz icin, ve de, 113 icin gecis izni olmadigi nedeniyle, karsi makina belli bi sure 113'e baglanmaya calisacak bize de telnet baglantisini bir sure acmayacaktır. Bu nedenle, biz, bu makinalara portun unreachable oldugunu bildiriyoruz ki, o makina da time-out'a dusmeyi beklemeyen, o portun kapali oldugunu ogrenip, denemekten vazgecek, bize baglantiyi derhal aciyor.

```
block in quick on fxp0 from 192.168.0.0/16 to any
block in quick on fxp0 from 172.16.0.0/12 to any
block in quick on fxp0 from 10.0.0.0/8 to any
block in quick on fxp0 from 127.0.0.0/8 to any
block in quick on fxp0 from any to 192.168.0.0/32
block in quick on fxp0 from any to 192.168.0.255/32
block in quick on fxp0 from any to 172.16.0.0/32
block in quick on fxp0 from any to 172.16.255.255/32
block out quick on fxp0 from any to 192.168.0.0/16
block out quick on fxp0 from any to 172.16.0.0/12
block out quick on fxp0 from any to 10.0.0.0/8
block out quick on fxp0 from any to 127.0.0.0/8
```

Burda da yaptgimiz sey, het turlu non-routable private ip uzayindan, bizim networkumuze, bizim agimizdan da, disariya bu kabilden paketlerin cikmasini engelliyoruz. Bu, ip spoofing denen olayin bir nevi onune gecmek icin.

```
pass in quick on fxp0 proto icmp from any to 192.168.0.0/24 icmp-type 0
pass in quick on fxp0 proto icmp from any to 192.168.0.0/24 icmp-type 3
pass in quick on fxp0 proto icmp from any to 192.168.0.0/24 icmp-type 11
```

ICMP_ECHO_REPLY, 'a izin verelim ki, insanlar ping cekerek bizim up & alive oldugumuzu anlayabilsinler, icmp-type 11 ttl expired, 0, echo-reply, 3, destination-unreachable, icin oluyor arkadaslar. Bu hangisiydi hatirlamiyorum, galiba firewall rfc ile belirtilen standartlara uyum icin. yani belli internet servisleri calistiriyorsaniz, minumum echo_reply ve ttl expired'a izin vermek zorundasiniz. Ama arkada oyle, web/smtp/dns felam gibi servislerin yoksa buna da izin vermeyin, tavsiye etmiyorum, cunku bu kadarcik icmp bile, cok *evil* seyler icin kullanilabiliyor ! Firewall arkasinda calisan sql serveriniza icmp ASLA izni vermeyin mesela, insanların orda bir sql server var uzakta deyip, onunla yerli yersiz ugrasmasini istemeyiz degil mi?

```
pass in quick on fxp0 proto tcp from any to 193.140.192.223 port = 25 flags S keep state
pass in quick on fxp0 proto tcp from any to 193.140.192.40 port = 80 flags S keep state
pass in quick on fxp0 proto tcp from any to 193.140.192.223 port = 110 flags S keep state
```

Burda da, mail sunucumuza, 25 ve 110 (smtp ve pop3) port erisimini , web sunucumuza da 80 (www) flags s ve keep-state sihirlilicuklerini kullanarak izin veriyoruz.

FreeBSD IpFilter Howto

```
pass out quick on fxp0 proto tcp from any to any keep state
pass out quick on fxp0 proto udp from any to any keep state
pass out quick on fxp0 proto icmp from any to any keep state
```

Evet, bizim agimizdan disari butun cikisa izin! Evet, biliyorum guzel bi dizayn sayilmaz pek, ama simdilik bu kadar, ileri de vakit olursa bu detaylara da gireriz icabinda.

```
block return-rst in quick on fxp0 proto tcp from any to any
block return-icmp-as-dest(port-unr) in quick on fxp0 proto udp from any to any
block in log quick on fxp0 proto icmp from any to any
```

Diger butun tcp paketler icin, default olarak, tcp-reset paketi yolluyoruz. Udp paketler icin de, gercek destination adresi, pakete basarak (firewallin ip'sini degil) icmp port unreachable paketi yolluyoruz. icmp'leri de logluyoruz ;)

Evet arkadaşlar, simdi /etc/ipf.rules dosyamiz adam oldu sayilir. Simdi de, /etc/ipnat.rules dosyamiza bakalim.

/etc/ipnat.rules

Firewall teknolojilerininin buyul kiyaklarından biri de, hic suphesiz, bir suru bilgisayarın, Internet servis saglayiciların ruhu bile duymadan tek bir dis arayuz aracılığıyla internetle bulusabilmesidir. Yani bir suru bilgisayarın tek bir ip/internet baglantisi ile internete cikmasi. Linux dunyasi bunu, IP Masquerading olarak biliyor, ama aslindan bunun ASIL ismi, Network Address Translation yani NAT. ipfilter, bize ip'eri ip'lere `map` etmeyi olasi kiliyor. Aslinda, ipfilter, NPAT yapıyor, yani Network Port & Address Translation. Ipfilter sayesinde, kaynak ve hedef port ve adresi kolaylikla degistirebiliyoruz.

```
map fxp0 172.16.0.0/16 -> 193.140.192.219/32
```

Ne zaman ki fxp0 arayuzumuze 172.16.0.0 blogundan bir paket gelse, paket, kaynak adres ve portu belirttigimiz (193.140.192.219) adresininkiyle degistiriliyor, orijinal hedef port ve adres korunarak yollanıyor. Hatta, ipfilter, bu islemi yaparken, ayni zamanda bunu bir litede tutarki, gelen paketleri de karsilayabilsin, istenilen adrese geri yollasin. Fekaati, yukaridaki kuralin bir eksigi var. Ya biz internete acilan ehternet'imiziz Ip'sini bilmiyorsak, yani mesela baglantimiz dinamic point-to-point baglantisi ise? O zaman IP yerine 0/32 yazarsak, NAT, kurali uygularken, tun0 ya da herneyse o arayuzdeki IP'yi otomatik bulacak, ve 0/32 yerinde kullanacaktır!.

```
map tun0 172.16.0.0/16 -> 0/32
```

Simdi... yukaridaki kuralimiz hala eksik sayilir. Yukarida kaynak porta herhangi bir manipulasyon yapilmiyor, bu da problemlere yol acabilir. Mesela, 172.16.3.3 makinasi da, 172.16.3.9 makinasi da, kaynal portu 1300 olan bir paket yolladi. E simdi, dis arayuzumuzden ayni IP'den iki ayni port disari cikacak? Luckily, durum boyle degil, ifilter `portmap` kelimesiyle portlari da kendisinin yonetmesine imkan sagliyor.

```
map tun0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:30000
```

FreeBSD IpFilter Howto

Evet, simdi soyle bir senaryo dusunelim: 193.140.192.40 Ip'sinde, 80. portta calisan bir IIS webserverimiz var. `.com` dunyasinda, makinamiz slk slk rahatsiz edilecektir... Biz bunu `prilileged port` lar olan 0-1024 arasinda bir portta degil de, port > 1024 calistirmak istiyoruz. Mesela 8000. portta. E ama, insanlar standart port olan 80'e baglanmak isteyeceklerdir. Insanlar bizim web serverimizin 8000. portta oldugunu hangi cehennemden ogrenecekler? Isin asli oyle degil, ipfilter, bize `redirection` sayesinde bunu gerceklestirebilme imkanini sagliyor:

```
rdr fxp0 193.140.205.219/32 port 80 -> 192.168.0.5 port 8000
```

Dis arayuzumuz olan fxp0'da 193.140.205.219/32 IP'sine 80. portuna gelen paketler, firewall'imizin arkasindaki 192.168.0.5 makinasinin 8000. portuna yonlendiriliyor. Temiz is, degil mi?

ipnat, paketler uzerinden gecerken, paketlerin tekrar yazimina olanak sagladigi icin, `application proxy` programinin uygulanabilmesi icin cok guzel bir mekan oluyor. Bu da, ftp gibi bazi protokollerin NAT'inda, mudahil olmayi sagliyor. Bu, da cogu ftp server'in `active connection` tutmasindan dolayi, ortaya cikabilecek problemleri onluyor:

```
map fxp0 0/0 -> 193.140.192.219/32 proxy port ftp ftp/tcp
```

Fakat ftp ile ilgili yukaridaki kuralimizin butun diger NAT kurallarindan once yazilma zorunlulugu vardir. Secure LAN'imizi da internet'e cikaralim:

```
map fxp0 192.168.0.0/24 -> 193.140.192.219/32 portmap tcp/udp 12500:60000  
map fxp0 192.168.0.0/24 -> 193.140.192.219/32
```

evet, simdi de,

```
pathfinder:~# ipnat -CF -f /etc/ipnat.rules
```

ile ipnat kurallarimizi etkin hale getiriyoruz.

Yardimci programlar: ipmon, ipfstat

IPFSTAT:

Ipstat'i en basit formunda calistirdiginizde, firewall'inizin neler yaptigini gormenizi saglayacak istatistiki bilgiler verecektir.

```
pathfinder:~# ipfstat  
input packets: blocked 99286 passed 1255609 nomatch 14686 counted 0  
output packets: blocked 4200 passed 1284345 nomatch 14687 counted 0
```

FreeBSD IpFilter Howto

```
input packets logged: blocked 99286 passed 0
output packets logged: blocked 0 passed 0
packets logged: input 0 output 0
log failures: input 3898 output 0
fragment state(in): kept 0 lost 0
fragment state(out): kept 0 lost 0
packet state(in): kept 169364 lost 0
packet state(out): kept 431395 lost 0
ICMP replies: 0 TCP RSTs sent: 0
Result cache hits(in): 1215208 (out): 1098963
IN Pullups succeeded: 2 failed: 0
OUT Pullups succeeded: 0 failed: 0
Fastroute successes: 0 failures: 0
TCP cksum fails(in): 0 (out): 0
Packet log flags set: (0)
none
```

Eger, bu listeyi, her kuralimizin neler yaptigini `hit-count` lariyla detayli olarak gormek istersek `-hio` parametresini ekliyoruz:

```
pathfinder:~# ipfstat -hio
2451423 pass out on fxp0 from any to any
354727 block out on ppp0 from any to any
430918 pass out quick on ppp0 proto tcp/udp from 193.140.205.0/24 to any keep state keep frags
```

Hatirlarsaniz `/etc/ipf.rules` 'lari yazarken magic state table' dan bahsetmistim ;) Iste state table'in su andaki durumunu gormek isterseniz:

```
pathfinder:~# ipfstat -s
281458 TCP
319349 UDP
0 ICMP
19780145 hits
5723648 misses
0 maximum
0 no memory
1 active
319349 expired
281419 closed
100.100.100.1 -> 193.140.205.219 ttl 864000 pass 20490 pr 6 state 4/4
pkts 196 bytes 17394 987 -> 22 585538471:2213225493 16592:16500
pass in log quick keep state
pkt_flags & b = 2, pkt_options & ffffffff = 0
pkt_security & ffff = 0, pkt_auth & ffff = 0
```

IPMON:

ipfstat, bize yukaridaki turden accounting bilgisi saglamasi acisindan gercekten cok guzel bir program, ama bazen bundan daha fazlasini isteyebiliyoruz. Yani, gelen giden paketlerin loglanmasini isteyebiliyoruz. Iste ipmon bunun icin yazilmis bi tool. Eger, `state-table`'imizi calisirken gormek istersek:

```
pathfinder:~# ipmon -o S
01/08/1999 15:58:57.836053 STATE:NEW 100.100.100.1,53 -> 193.140.205.15,53 PR udp
01/08/1999 15:58:58.030815 STATE:NEW 193.140.205.15,123 -> 128.167.1.69,123 PR udp
01/08/1999 15:59:18.032174 STATE:NEW 193.140.205.15,123 -> 128.173.14.71,123 PR udp
01/08/1999 15:59:24.570107 STATE:EXPIRE 100.100.100.1,53 -> 193.140.205.15,53 PR udp Pkts 4 Bytes 356
01/08/1999 16:03:51.754867 STATE:NEW 193.140.205.13,1019 -> 100.100.100.10,22 PR tcp
01/08/1999 16:04:03.070127 STATE:EXPIRE 193.140.205.13,1019 -> 100.100.100.10,22 PR tcp Pkts 63 Bytes
4604
```

FreeBSD IpFilter Howto

Hanimi\$ NAT table? Iste burda :

```
pathfinder:~# ipmon -o N
01/08/1999 05:30:02.466114 @2 NAT:RDR 193.140.205.253,113 <- -> 193.140.205.253,113
[100.100.100.13,45816]
01/08/1999 05:30:31.990037 @2 NAT:EXPIRE 193.140.205.253,113 <- -> 193.140.205.253,113
[100.100.100.13,45816] Pkts 10 Bytes 455
```

Simdi, hepsini bir araya koyalım:

----- /etc/ipf.rules buradan basliyor

pass out quick on lo0
pass in quick on lo0

block in quick on fxp0 all with opt lsrr
block in quick on fxp0 all with opt ssrr
block in quick on fxp0 proto tcp from any to any flags FUP

block return-icmp(port-unr) in quick on fxp0 from any to any port = 113

```
##
## Default is block all, so i didn't specifically blocked icmp
##
## block non-routable for spoofing protection --
block in quick on fxp0 from 192.168.0.0/16 to any
block in quick on fxp0 from 172.16.0.0/12 to any
block in quick on fxp0 from 10.0.0.0/8 to any
block in quick on fxp0 from 127.0.0.0/8 to any
block in quick on fxp0 from any to 192.168.0.0/32
block in quick on fxp0 from any to 192.168.0.255/32
block in quick on fxp0 from any to 172.16.0.0/32
block in quick on fxp0 from any to 172.16.255.255/32
block out quick on fxp0 from any to 192.168.0.0/16
block out quick on fxp0 from any to 172.16.0.0/12
block out quick on fxp0 from any to 10.0.0.0/8
block out quick on fxp0 from any to 127.0.0.0/8
```

```
##
```

FreeBSD IpFilter Howto

```
## !!!! DANGER icmp types 0, 11 allowed to DMZ
##
pass in quick on fxp0 proto icmp from any to 192.168.0.0/24 icmp-type 0
pass in quick on fxp0 proto icmp from any to 192.168.0.0/24 icmp-type 11

## for web server
pass in quick on fxp0 proto tcp from any to 192.168.0.1 port = 80 flags S keep state

## for smtp servers
pass in quick on fxp0 proto tcp from any to 192.168.0.5 port = 25 flags S keep state
pass in quick on fxp0 proto tcp from any to 192.168.0.3 port = 25 flags S keep state

## pop3 for smtp servers
pass in quick on fxp0 proto tcp from any to 192.168.0.3 port = 110 flags S keep state
pass in quick on fxp0 proto tcp from any to 192.168.0.5 port = 110 flags S keep state

## ssh sadece apache.cslab.itu.edu.tr 'den
pass in quick on fxp0 proto tcp from 160.75.80.171/32 to 192.168.0.11 port = 22 flags S keep state

## imap for only our localnet
pass in quick on fxp0 proto tcp from 193.140.205.0/24 to 192.168.0.5 port = 143 flags S keep state

## mssql server port unu uzaktaki bir makina icin acmamizi istemisler:
pass in quick on fxp0 proto tcp from 193.140.205.55/32 to 192.168.0.43 port = 1433 flags S keep state

## disari cikisa kayitsiz sartsiz izin veriyoruz:
pass out quick on fxp0 proto tcp from any to any keep state
pass out quick on fxp0 proto udp from any to any keep state
pass out quick on fxp0 proto icmp from any to any keep state

## diger butun tcp paketler icin by-default tcp reset paketi yolluyoruz.
block return-rst in quick on fxp0 proto tcp from any to any

## icmp'ler icin de port unreachable, hem de destination adresi source adres yaparak:
block return-icmp-as-dest(port-unr) in quick on fxp0 proto udp from any to any

## icmp'leri de bloke edip logluyoruz.
block in log quick on fxp0 proto icmp from any to any

----- /etc/ipf.rules burada bitiyor -----
----- /etc/ipnat.rules buradan basliyor -----
#
# See /usr/share/ipf/nat.1 for examples.
# edit the ipnat= line in /etc/rc.conf to enable Network Address Translation
#map ppp0 10.0.0.0/8 -> ppp0/32 portmap tcp/udp 10000:20000

## wishmaster ipnat.rules
## all networks ftp transaction
map fxp0 172.16.0.0/16 -> 193.140.192.219/32 proxy port ftp ftp/tcp
```

FreeBSD IpFilter Howto

```
## map pathfinder to 212.174.107.33
map fxp0 172.16.3.3/32 -> 193.140.205.33/32 portmap tcp/udp 12500:60000
map fxp0 172.16.3.3/32 -> 193.140.205.33/32

## map secure lan to 193.140.192.219
map fxp0 192.168.0.0/24 -> 193.140.192.219/32 portmap tcp/udp 12500:60000
map fxp0 192.168.0.0/24 -> 193.140.192.219/32

## map local lan to 107.13
map fxp0 172.16.0.0/16 -> 193.140.192.219/32 portmap tcp/udp 12500:60000
map fxp0 172.16.0.0/16 -> 193.140.192.219/32

## Squid Redirection
#rdr fxp1 0/0 port 80 -> 172.16.0.1 port 8080 tcp

## for web server
rdr fxp0 193.140.205.219 port 80 -> 192.168.0.1 port 80

## for smtp servers
rdr fxp0 193.140.205.219 port 25 -> 192.168.0.5 port 25
rdr fxp0 193.140.205.220 port 25 -> 192.168.0.3 port 25

## pop3 for smtp servers
rdr fxp0 193.140.205.219 port 110 -> 192.168.0.5 port 110
rdr fxp0 193.140.205.220 port 110 -> 192.168.0.3 port 110

## ssh sadece apache.cslab.itu.edu.tr 'den
rdr fxp0 193.140.205.219 port 22 -> 192.168.0.11 port 22

## imap for only our localnet
rdr fxp0 193.140.205.220 port 143 -> 192.168.0.5 port143

## mssql server port unu uzaktaki bir makina icin acmamizi istemisler:
rdr fxp0 193.140.205.134 port 1433 -> 192.168.0.43 port 1433
```

----- /etc/ipnat.rules burada bitiyor -----

Bu iki dosyayi, kaydedip makinayi restart etmeniz, ya da restart etmemek istiyorsanız,

```
pathfinder:~# ipf -Fa -f /etc/ipf.rules; ipnat -CF -f /etc/ipnat.rules
```

girmeniz yeterli olacaktır.

SSS (SIK Sorulan Sorular)

◆ **Ipfilter calisan firewall'um bir sure sonra hicbir paketi handle etmiyor. Fakat makinami reboot edince hersey duzeliyor. Neden?**

State table ile alakali bir sorun. Bir sure sonra state table tutabilecegi "active" maximum state sayisina ulasip, dolunca artik "state" gerektiren hic bir baglantiyi route etmeyecektir. Bu durumda reboot etmek bir cozum oldugu gibi, reboot etmeden, **ipf -FS** komutunu verip, state table'i flush edebilirsiniz. Bu durumda olup olmadiginizi ipfstat -s komutunda maximum satirinin 0'dan buyuk bir deger oldugunu gorerek anlayabilirsiniz:

```
su-2.04# ipfstat -s
IP states added:
212339 TCP
41634 UDP
6 ICMP
11753795 hits
873319 misses
684 maximum
0 no memory
146 bkts in use
299 active
41638 expired
212042 closed
su-2.04#
```

Bu sorunu cozmek icin, STATE tablonuzun buyuklugunu artirmaniz lazim. /sys/netinet/ip_state.h dosyasinda soyle bir kisim var:

```
#define IPSTATE_SIZE 5737
#define IPSTATE_MAX 4013
```

Bu degerler, fw'nuzu evde kullanmak icin gerekli degerler :) Bunlari universite ortaminda kullanmak gibi bir dusunceniz varsa, bu rakamlari kesinlikle artirmaniz lazim. Yalniz artirirken de, (hashing methodlarinin gerektirdigi uzre) **IPSTATE_MAX**, **IPSTATE_SIZE**'in yaklasik olarak **%70**'i olacak, ve bu heriki deger de **asal sayi** olacak sekilde artirin. Sonra yeni kernel derleyip, makinayi reboot ediniz.

◆ **Ipfilter, ticari bir isyerini koruyabilecek kadar "mature" mu?**

Evet, senelerdir cok tecrubeli bir programci olan Darren tarafından gelistiriliyor ve basta Lucent olmak uzere, Cisco'da dahil, bazi sirketler ipfilter kodunu programlarinda kullaniyorlar.

◆ **Ipfilter, bir T3'u koruyabilir mi?**

Evet, yukarida anlattigimiz ipstate size la alakali ayarlamayi yapar, cok rule'un sozkonusu oldugu yerlerde rule group kullanirsaniz, "cok kolaylikla" bir saturated T3 baglantiyi handle edebilir. 2 Mbit/sec(E1) baglantinin oldugu 3 universitede su an kullaniliyor.

◆ **Ipfilter icin bir GUI var mi?**

Evet, hem de gayet basarili: <http://inc2.com/isba/> FW1'i animsatiyor...

TODO:

- FTP sunucularin filtrelenmesi
- fragment-filtering
- Performance Tuning (State Table ile alakali)
- Scalable ipfilter ozellikleri
- Round-robin redirection

Kaynaklar:

Official Ipf-Homepage:

<http://coombs.anu.edu.au/~avalon/ip-filter.html>

Ipf-Howto:

<http://www.obfuscation.org/ipf/>

Linux Firewall Howto

OpenBSD Networking FAQ

<http://www.enderunix.org/openbsd/faq/faq6.html>

Son Sozler:

Lutfen soru sormadan once bu dokumani bir daha okuyun Eger sorunuza cevap bulamazsaniz false.net/ipfilter adresindeki mail arshivlerini okuyun. Eger hala aradiginizi bulamazsaniz !!! bana mail atin belki bir kac ay icinde cevap verebilirim.

murat@EnderUNIX.ORG

Murat Balaban

<http://www.EnderUNIX.ORG/>

<http://www.enderunix.org/murat/>

Copyright (c) 2001 Murat Balaban Kaynak gosterilmek sartıyla kullanilabilir.

EnderUNIX Software Development Team Member